

Best Practices for Alert management



Ideally, alerting should be both sufficient (catching all conditions that need attention) and noise-free (only alerting on conditions that need attention.) To achieve this, you should be continually tuning away non-meaningful alerts. An equally important activity is to not close any incident that was not alerted on until the alerts to detect it have been created.

The steps below can help you achieve this state with your LogicMonitor deployment.

Prevent alerts

Use Scheduled Downtime

If you are going to be working on a system, you should schedule downtime so that alerts are not escalated during the expected period of work. Scheduled Downtime (SDT) can be scheduled on an individual instance (e.g. a single drive); a host's datasource (e.g. all the drives on the host), a host (all monitoring on a host), or on a group level (either all alerts for all hosts in a group, or a single datasource for all hosts in a group).

If you have regular maintenance windows, you can schedule recurrent SDT's - daily, weekly, or monthly.

Generally you should set the SDT on the narrowest scope possible (instance, or host, or group), to ensure that other valid alerts not related to the work are not suppressed.

Route Alerts

Alert escalation and routing can be as flexible as you wish, but, as in other areas, simplicity is preferred.

A recommended system is to define two escalation chains for each department - one for warning alerts that are emailed only and does not escalate (i.e. a resend interval of zero), and one for errors and critical alerts that are sent to pagers and emails, escalating among staff members.

Then define rules that match on group (or hostname pattern) and severity - a warning rule that routes warnings to the email chain; and rule that routes all other alerts (which will be errors and criticals) to the pager chain.

Name	Priority	Level	Group	Host	Datasource	Datapoint	Resend (escalation) interval	Escalation Chain
ProductionServersWarning	1	Warn	*	prod*	*	*	0	Prod Warnings Stage 1 - lmsupport
ProdServersErrors	2	All	*	prod*	*	*	20	ProdErrors Stage 1 - 80569807 Stage 2 - 65028309

This model can be repeated for multiple departments, routing the alerts for each department to the correct destinations.

It can also be useful to put in rules that route some alerts to chains with no destinations - such alerts are only visible in the web UI. (e.g. rules for some QA systems.)

Deal with Alerts

Once an alert comes in, these are the recommended actions:

Respond to the Alert

For each alert, you should respond by acknowledging the alert (reply to the alert email or SMS with a line “Ack I am investigating”, for example), or scheduling downtime for the alert for a period of time (reply with “SDT 1” to schedule 1 hour of downtime for this alert.) Both responses will stop escalation of the alert to the next stage in the escalation chain, and notify all contacts that the alert has been sent to.

You can also immediately escalate to the next stage (by replying to an alert with “Next”).

SDT or Ack?

Generally, you should acknowledge an alert if you believe you will be able to resolve the condition, or you will tune the alert thresholds, disable the alert or otherwise clear the alert.

If you expect to deal with the alert later (for example, if it is a non-urgent issue that you were alerted about in the middle of the night, but want to deal with in the morning), or if you expect the alert to recur repeatedly before you will be able to address the underlying root cause (for example, you know a server needs more memory, triggering memory usage alerts, but the memory will not arrive for a week), then you should schedule downtime for the alert to expire at the time you expect the issue to be resolved. This has the advantage that if you have not resolved the underlying issue by the time the scheduled downtime expires, you will be alerted again about the issue.

An acknowledgement only effects the current alert occurrence, and is in effect until the condition clears (so if the alert condition clears, then triggers again, a new alert is sent). SDT suppresses all alert escalation for current and future alerts until the SDT expires, so it may affect several alert occurrences.

Analyse the Alert

Now a decision needs to be made - was this alert appropriate?

If it was a real issue, and delivered by the appropriate mechanism to the right people, then the response is to simply resolve the underlying issue. e.g. If the alert was that a production server was swapping at a high rate, and it was sent via page to the right people, the alerting was correct - the correct action is to cure the memory issue on the server (probably by restarting a process, adding more memory, or moving workloads)

If the alert was regarding a real issue, but escalated incorrectly - to the wrong people, or with the wrong level of urgency (e.g. via pager for an issue that was not urgent) - then the action should be to adjust the alert escalations. This can be done by changing the severity of the alert (among warning, error, and critical levels - this is done by adjusting the thresholds that are triggered), or by adjusting the alert rules and escalation chains.

If the alert was as a result of expected actions, ensure that SDT is scheduled appropriately next time, and put in policies to require SDT actions before maintenance, or create recurring SDT periods.

If the alert was a noise alert - not one you wish to receive in the future - there are a variety of ways to prevent it:

- [adjusting the thresholds](#), globally, for a group, a host or an instance. (e.g. Changing the threshold for spare disk alerting to 1 for small NetApps.) Note that you can also use time based thresholds to apply different thresholds during different periods of time prevent alerts (e.g. for CPU load on NetApps during weekly disk scrubs).
- disabling all alerts for a [group, a host](#) or a [specific instance](#). (e.g. you may not want alerts on development machines, so you disable the alerts for that group of hosts.)
- disabling the datasource data collection globally, for a group, a host or a specific instance. (e.g. disabling the collection of all Apache data on development machines.)
- eliminate the alerting datasource instance from discovery, or create a cloned datasource to discover/classify it with different alert thresholds. If there is a set of instances that should not be discovered (e.g. NFS mounted filesystems on linux hosts) or that you want to have different thresholds than other instances (e.g. QA VIPs on load balancers), you can achieve that with Active Discovery filtering.

When there was no alert

No issue should be considered resolved if monitoring will not detect its recurrence.

Even with good monitoring, outages will occur. Best practices dictate that an issue not be considered resolved until monitoring is in place to detect the root cause, or provide earlier warning if possible. Graphs should be investigated to look for behavior that can be used to alert about the event. For example, if a Java application experiences a service affecting outage due to a large number of users overloading the system, the earliest warning of an impending issue may be an increase in the number of busy threads, which can be tracked via JMX monitoring. An alert threshold should be placed on this metric, to give advance warning before the next event, which could allow time to add another system to share the load, or activate load shedding

mechanisms, and so on.

This is a very important principle – just because things are working again, it does not mean issues should be closed unless you are happy with the warning your monitoring gave about the issue before it started, or the kind of alerts and alert escalations that occurred during the issue. It's possible that the issue is one with no way to warn in advance (for example, sudden panic of a system), but this process of evaluation should be undertaken for every service impacting event.

Practices to avoid Alert overload

If you have too many noisy alerts, that go off too frequently, people will tune them out – then when you get real, service impacting alerts, they will be tuned out, too. Like the critical production service outage alerts put into scheduled downtime for 8 hours, as the admin assumed it was “another false alert” and went back to sleep. How to prevent this?

- adopt sensible escalation policies for your organization. Distinguish between warnings (that admins should be aware of, but do not require immediate actions) and error or critical level alerts, that require pager notifications. (No need to awaken people if NTP is out of synchronization – but if the primary database volume is experiencing 200 ms latency for read requests from its disk storage, and end user transaction time is now 18 seconds, then all hands on deck).
- route the right set of alerts to the right group of people. There is no point in the DBA being alerted about network issues that should have gone to the networking group, or vice versa.
- make sure you tune your thresholds appropriately. Every alert should be real and meaningful. If any alerts are ‘false positives’ (such as alerts about QA systems being down), tune the monitoring.
- If an alert is triggered, but everything is working fine - use it as an indication of something to investigate. But if the investigation shows no issues in your environment, don't hesitate to adjust the default threshold, or disable that alert. (Also feel free to contact support at LogicMonitor, to discuss the ramifications of changing or disabling the alert. It was put there for a reason, but every environment is different.)
- ensure alerts are acknowledged, dealt with, and cleared. You don't want to see hundreds of alerts on your monitoring system. For large enterprises with many hosts, use alert filtering to see a filtered view of just the groups of systems you are responsible for, allowing focus.
- periodically sort alerts by duration, and focus on cleaning out those that have been in alert for longer than a day.
- create a weekly alert report that lists all alerts for the last week, delivered via email to your department, then meet to review the top alerts, by host, or by alert type. Investigate to see whether there are issues in monitoring, or the systems, or operational processes, that can reduce the frequency of these alerts.
- use alert trend reports to track which hosts/groups are getting the most alerts

And as always, don't hesitate to contact LogicMonitor support if you have questions.